Language-integrated Provenance

Stefan Fehrenbach

James Cheney

PPDP 2016

lfcs | Laboratory for Foundations of Computer Science

THE UNIVERSITY of EDINBURGH informatics

A database

Agencies

oid	name	based_in	phone
1	EdinTours	Edinburgh	8740 2489 123
2	Burns's	Glasgow	9307 2394 104

ExternalTours

oid	name	destination	type	price in £
3	EdinTours	Edinburgh	bus	20
4	EdinTours	Loch Ness	bus	50
5	EdinTours	Loch Ness	boat	200
6	EdinTours	Firth of Forth	boat	50
7	Burns's	Islay	boat	100
8	Burns's	Mallaig	train	40

Language-integrated query

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (a.name == e.name
        && e.type == "boat")
    [(name = e.name,
      phone = a.phone)]
}
```

Agencies

oid	name	based_in	phone
1	EdinTours	Edinburgh	8740 2489 123
2	Burns's	Glasgow	9307 2394 104

ExternalTours

oid	name	destination	type	price in £
3	EdinTours	Edinburgh	bus	20
4	EdinTours	Loch Ness	bus	50
5	EdinTours	Loch Ness	boat	200
6	EdinTours	Firth of Forth	boat	50
7	Burns's	Islay	boat	100
8	Burns's	Mallaig	train	40

Language-integrated query

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (a.name == e.name
        && e.type == "boat")
    [(name = e.name,
      phone = a.phone)]
}
```

name	phone
EdinTours	8740 2489 123
EdinTours	8740 2489 123
Burns's	9307 2394 104

Agencies

oid	name	based_in	phone
1	EdinTours	Edinburgh	8740 2489 123
2	Burns's	Glasgow	9307 2394 104

ExternalTours

oid	name	destination	type	price in £
3	EdinTours	Edinburgh	bus	20
4	EdinTours	Loch Ness	bus	50
5	EdinTours	Loch Ness	boat	200
6	EdinTours	Firth of Forth	boat	50
7	Burns's	Islay	boat	100
8	Burns's	Mallaig	train	40

Where-provenance

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (a.name == e.name
        && e.type == "boat")
    [(name = e.name,
      phone = a.phone)]
}
```

name	phone
EdinTours	8740 2489 123
EdinTours	8740 2489 123
Burns's	9307 2394 104

Agencies

oid	name	based_in	phone
1	EdinTours	Edinburgh	8740 2489 123
2	Burns's	Glasgow	9307 2394 104

ExternalTours

oid	name	destination	type	price in £
3	EdinTours	Edinburgh	bus	20
4	EdinTours	Loch Ness	bus	50
5	EdinTours	Loch Ness	boat	200
6	EdinTours	Firth of Forth	boat	50
7	Burns's	Islay	boat	100
8	Burns's	Mallaig	train	40

Where-provenance

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (a.name == e.name
        && e.type == "boat")
    [(name = e.name,
      phone = a.phone)]
}
```

name	phone
EdinTours	8740 2489 123
EdinTours	8740 2489 123
Burns's	9307 2394 104

Agencies

oid	name	based_in	phone
1	EdinTours	Edinburgh	8740 2489 123
2	Burns's	Glasgow	9307 2394 104

ExternalTours

oid	name	destination	type	price in £
3	EdinTours	Edinburgh	bus	20
4	EdinTours	Loch Ness	bus	50
5	EdinTours	Loch Ness	boat	200
6	EdinTours	Firth of Forth	boat	50
7	Burns's	Islay	boat	100
8	Burns's	Mallaig	train	40

Where-provenance

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (a.name == e.name
        && e.type == "boat")
    [(name = e.name,
      phone = a.phone)]
}
```

name	phone
EdinTours	8740 2489 123
EdinTours	8740 2489 123
Burns's	9307 2394 104

Agencies

oid	name	based_in	phone
1	EdinTours	Edinburgh	8740 2489 123
2	Burns's	Glasgow	9307 2394 104

ExternalTours

oid	name	destination	type	price in £
3	EdinTours	Edinburgh	bus	20
4	EdinTours	Loch Ness	bus	50
5	EdinTours	Loch Ness	boat	200
6	EdinTours	Firth of Forth	boat	50
7	Burns's	Islay	boat	100
8	Burns's	Mallaig	train	40

Where-provenance

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (a.name == e.name
        && e.type == "boat")
    [(name = e.name,
      phone = a.phone)]
}
```

name	phone
EdinTours	8740 2489 123
EdinTours	8740 2489 123
Burns's	9307 2394 104

Agencies

oid	name	based_in	phone
1	EdinTours	Edinburgh	8740 2489 123
2	Burns's	Glasgow	9307 2394 104

ExternalTours

oid	name	destination	type	price in £
3	EdinTours	Edinburgh	bus	20
4	EdinTours	Loch Ness	bus	50
5	EdinTours	Loch Ness	boat	200
6	EdinTours	Firth of Forth	boat	50
7	Burns's	Islay	boat	100
8	Burns's	Mallaig	train	40

Lineage (why-provenance)

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (a.name == e.name
        && e.type == "boat")
    [(name = e.name,
      phone = a.phone)]
}
```

name	phone
EdinTours	8740 2489 123
EdinTours	8740 2489 123
Burns's	9307 2394 104

Agencies

oid	name	based_in	phone
1	EdinTours	Edinburgh	8740 2489 123
2	Burns's	Glasgow	9307 2394 104

ExternalTours

oid	name	destination	type	price in £
3	EdinTours	Edinburgh	bus	20
4	EdinTours	Loch Ness	bus	50
5	EdinTours	Loch Ness	boat	200
6	EdinTours	Firth of Forth	boat	50
7	Burns's	Islay	boat	100
8	Burns's	Mallaig	train	40

Lineage (why-provenance)

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (a.name == e.name
        && e.type == "boat")
    [(name = e.name,
      phone = a.phone)]
}
```

name	phone
EdinTours	8740 2489 123
EdinTours	8740 2489 123
Burns's	9307 2394 104

Agencies

oid	name	based_in	phone
1	EdinTours	Edinburgh	8740 2489 123
2	Burns's	Glasgow	9307 2394 104

ExternalTours

oid	name	destination	type	price in £
3	EdinTours	Edinburgh	bus	20
4	EdinTours	Loch Ness	bus	50
5	EdinTours	Loch Ness	boat	200
6	EdinTours	Firth of Forth	boat	50
7	Burns's	Islay	boat	100
8	Burns's	Mallaig	train	40

Language-integrated provenance builds on

Language-integrated query

- LINQ: … *Meijer, Beckman, Bierman*. SIGMOD 2006
- The script-writer's dream. *Cooper*. DBPL 2009
- Query shredding: … *Cheney, Lindley, Wadler*. SIGMOD 2014
- Effective quotation: … *Cheney, Lindley, Radanne, Wadler*. PEPM 2014

Provenance in databases

- Why and where: … *Buneman, Khanna, Tan*. ICDT 2001
- On the expressiveness of implicit provenance … *Buneman, Cheney, Vansummeren*. TODS 2008
- Perm: … *Glavic, Alonso*. ICDE 2009
- Using SQL for efficient generation and querying … *Glavic, Miller, Alonso*. Buneman Festschrift 2013

This talk

1. Why?
2. Language-integrated where-provenance in Links^{W}
3. Rewriting Links^{W} to Links

The paper

4. User-defined where-provenance
5. Lineage in Links^{L} and its translation to Links
6. Performance

Why?

Easy access to data *and its provenance*

Provenance is not data – it is metadata
- data without provenance is less than complete
- provenance on its own is quite useless
- data with fake provenance is an affront

Calculating provenance and propagating it manually is hard
- or least cumbersome enough to want to automate it

Where-provenance in Links^{W}

Mark data carrying provenance metadata with an abstract type:

Prov(O) O is a base type

Two operations:

$$\frac{\Gamma \vdash M : \mathrm{Prov}(O)}{\Gamma \vdash \textbf{data}\ M : O} \qquad \frac{\Gamma \vdash M : \mathrm{Prov}(O)}{\Gamma \vdash \textbf{prov}\ M : (\mathrm{String}, \mathrm{String}, \mathrm{Int})}$$

No constructor! – only the runtime can create provenance-annotated data

Print as a comment, because it cannot appear in a program anyway:

"EdinTours" #("Agencies", "name", 2)

Language-integrated query in Links

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (a.name == e.name
         && e.type == “boat”)
    [(name = e.name,
      phone = a.phone)]
}
```

Language-integrated query in Links

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (a.name == e.name
         && e.type == “boat”)
    [(name = e.name,
      phone = a.phone)]
}
```

```
var agencies =  table “Agencies”
                with (oid: Int,
                      name: String,
                      based_in: String,
                      phone: String)
```

Language-integrated query in Links

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (a.name == e.name
         && e.type == “boat”)
    [(name = e.name,
      phone = a.phone)]
}
```

```
var agencies =  table “Agencies”
                with (oid: Int,
                      name: String,
                      based_in: String,
                      phone: String)
```

```
agencies : [(oid: Int,
             name: String,
             based_in: String,
             phone: String)]
```

Language-integrated query in Links

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (a.name == e.name
         && e.type == "boat")
    [(name = e.name,
      phone = a.phone)]
}
```

```
[(name = "EdinTours",
  phone = "8740 2489 123"),
 (name = "EdinTours",
  phone = "8740 2489 123"),
 (name = "Burns's",
  phone = "9307 2394 104")]

 : [(name: String, phone: String)]
```

```
var agencies =  table "Agencies"
                with (oid: Int,
                      name: String,
                      based_in: String,
                      phone: String)
```

```
agencies : [(oid: Int,
             name: String,
             based_in: String,
             phone: String)]
```

Where-provenance in Links^{W}

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (a.name == e.name
         && e.type == "boat")
    [(name = e.name,
      phone = a.phone)]
}
```

```
var agencies =  table "Agencies"
                with (oid: Int,
                      name: String,
                      based_in: String,
                      phone: String)
```

```
[(name = "EdinTours",          #("ExternalTours", "name", 5)
  phone = "8740 2489 123"),    #("Agencies", "phone", 1)
 (name = "EdinTours",          #("ExternalTours", "name", 6)
  phone = "8740 2489 123"),    #("Agencies", "phone", 1)
 (name = "Burns's",            #("ExternalTours", "name", 7)
  phone = "9307 2394 104")]    #("Agencies", "phone", 2)
```

```
agencies : [(oid: Int,
             name: String,
             based_in: String,
             phone: String)]
```

: [(name: Prov(String), phone: Prov(String))]

Where-provenance in Links^{W}

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (a.name == e.name
         && e.type == "boat")
    [(name = e.name,
      phone = a.phone)]
}
```

```
var agencies =  table "Agencies"
                with (oid: Int,
                      name: String,
                      based_in: String,
                      phone: String)
                where phone prov default,
                      name prov default
```

```
[(name = "EdinTours",          #("ExternalTours", "name", 5)
  phone = "8740 2489 123"),    #("Agencies", "phone", 1)
 (name = "EdinTours",          #("ExternalTours", "name", 6)
  phone = "8740 2489 123"),    #("Agencies", "phone", 1)
 (name = "Burns's",            #("ExternalTours", "name", 7)
  phone = "9307 2394 104")]    #("Agencies", "phone", 2)
```

```
agencies : [(oid: Int,
             name: String,
             based_in: String,
             phone: String)]
```

```
 : [(name: Prov(String), phone: Prov(String))]
```

Where-provenance in Links^{W}

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (a.name == e.name
         && e.type == "boat")
    [(name = e.name,
      phone = a.phone)]
}
```

```
var agencies =  table "Agencies"
                with (oid: Int,
                      name: String,
                      based_in: String,
                      phone: String)
                where phone prov default,
                      name prov default
```

```
[(name = "EdinTours",          #("ExternalTours", "name", 5)
  phone = "8740 2489 123"),    #("Agencies", "phone", 1)
 (name = "EdinTours",          #("ExternalTours", "name", 6)
  phone = "8740 2489 123"),    #("Agencies", "phone", 1)
 (name = "Burns's",            #("ExternalTours", "name", 7)
  phone = "9307 2394 104")]    #("Agencies", "phone", 2)

 : [(name: Prov(String), phone: Prov(String))]
```

```
agencies : [(oid: Int,
             name: Prov(String),
             based_in: String,
             phone: Prov(String))]
```

Where-provenance in Links^{W}

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (data a.name == e.name
         && e.type == "boat")
    [(name = e.name,
      phone = a.phone)]
}
```

```
var agencies =  table "Agencies"
                with (oid: Int,
                      name: String,
                      based_in: String,
                      phone: String)
                where phone prov default,
                      name prov default
```

```
[(name = "EdinTours",          #("ExternalTours", "name", 5)
  phone = "8740 2489 123"),    #("Agencies", "phone", 1)
 (name = "EdinTours",          #("ExternalTours", "name", 6)
  phone = "8740 2489 123"),    #("Agencies", "phone", 1)
 (name = "Burns's",            #("ExternalTours", "name", 7)
  phone = "9307 2394 104")]    #("Agencies", "phone", 2)

 : [(name: Prov(String), phone: Prov(String))]
```

```
agencies : [(oid: Int,
             name: Prov(String),
             based_in: String,
             phone: Prov(String))]
```

Where-provenance in Links^{W}

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (data a.name == data e.name
         && data e.type == "boat")
    [(name = e.name,
      phone = a.phone)]
}
```

```
var agencies =  table "Agencies"
                with (oid: Int,
                      name: String,
                      based_in: String,
                      phone: String)
                where phone prov default,
                      name prov default
```

```
[(name = "EdinTours",          #("ExternalTours", "name", 5)
  phone = "8740 2489 123"),    #("Agencies", "phone", 1)
 (name = "EdinTours",          #("ExternalTours", "name", 6)
  phone = "8740 2489 123"),    #("Agencies", "phone", 1)
 (name = "Burns's",            #("ExternalTours", "name", 7)
  phone = "9307 2394 104")]    #("Agencies", "phone", 2)

 : [(name: Prov(String), phone: Prov(String))]
```

```
agencies : [(oid: Int,
             name: Prov(String),
             based_in: String,
             phone: Prov(String))]
```

This talk

1. Why?
2. Language-integrated where-provenance in Links^{W}
3. Rewriting Links^{W} to Links

The paper

4. User-defined where-provenance
5. Lineage in Links^{L} and its translation to Links
6. Performance

Links^{W}

Prov(*O*)

data *M* **prov** *M*

table *N* **with** (a: A, …)
  where a **prov default**, …

for (x <-- *T*) *M*

update (x <-- *T*) *M*

Links

(data: *O*, prov: (String, String, Int))

M.data *M*.prov

Pair of table and view with initial provenance annotations

for (x <-- *T*.2()) *M*

update (x <-- *T.1*) *M*

table “Agencies”
with (oid: Int,
name: String,
based_in: String,
phone: String)
where phone **prov default**,
name **prov default**

```
table “Agencies”
with (oid: Int,
      name: String,
      based_in: String,
      phone: String)
where phone prov default,
      name prov default
```

```
for (a <-- table “Agencies” with …)
  [(oid = a.oid,
    name = (data = a.name,
            prov = (“Agencies”, “name”, a.oid)),
    based_in = a.based_in,
    phone = (data = a.phone,
             prov = (“Agencies”, “phone”, a.oid)))]
```

```
table “Agencies”
with (oid: Int,
      name: String,
      based_in: String,
      phone: String)
where phone prov default,
      name prov default
```

```
fun () {
  for (a <-- table “Agencies” with …)
     [(oid = a.oid,
       name = (data = a.name,
               prov = (“Agencies”, “name”, a.oid)),
       based_in = a.based_in,
       phone = (data = a.phone,
                prov = (“Agencies”, “phone”, a.oid)))]
}
```

```
table “Agencies”
with (oid: Int,
        name: String,
        based_in: String,
        phone: String)
where phone prov default,
        name prov default
```

```
(table “Agencies”
 with (oid: Int,
         name: String,
         based_in: String,
         phone: String),
 fun () {
  for (a <-- table “Agencies” with …)
     [(oid = a.oid,
       name = (data = a.name,
                 prov = (“Agencies”, “name”, a.oid)),
       based_in = a.based_in,
       phone = (data = a.phone,
                  prov = (“Agencies”, “phone”, a.oid)))]
 })
```

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (data a.name == data e.name
         && data e.type == “boat”)
    [(name = e.name,
      phone = a.phone)]
}
```

```
query {
  for (a <-- agencies)
  for (e <-- externalTours)
  where (data a.name == data e.name
         && data e.type == “boat”)
    [(name = e.name,
      phone = a.phone)]
}
```

```
query {
  for (a <- agencies.2())
  for (e <- externalTours.2())
  where (a.name.data == e.name.data
         && e.type.data == “boat”)
    [(name = e.name,
      phone = a.phone)]
}
```

Links^{W}

Prov(*O*)

data *M* **prov** *M*

table *N* **with** (a: A, …)
  where a **prov default**, …

for (x <-- *T*) *M*

update (x <-- *T*) *M*

Links

(data: *O*, prov: (String, String, Int))

M.data *M*.prov

Pair of table and view with initial provenance annotations

for (x <-- *T*.2()) *M*

update (x <-- *T.1*) *M*

This talk

1. Why?
2. Language-integrated where-provenance in Links^{W}
3. Rewriting Links^{W} to Links

The paper

4. User-defined where-provenance
5. Lineage in Links^{L} and its translation to Links
6. Performance

User-defined where-provenance

```
table “Agencies”
with (oid: Int,
        name: String,
        based_in: String,
        phone: String)
where phone prov default,
         name prov anyDBFun
```

```
fun () {
  for (a <-- table “Agencies” with …)
     [(oid = a.oid,
       name = (data = a.name,
                  prov = anyDBFun(a)),
       based_in = a.based_in,
       phone = (data = a.phone,
                   prov = defaultAgenciesPhone(a)))]
}
```

```
sig anyDBFun: (_) -> (String, String, Int)
fun anyDBFun (r) {
  (“Answers”, “Life, Universe and everything…”, 42)
}
```

```
sig defaultAgenciesPhone: (_) -> (String, String, Int)
fun defaultAgenciesPhone (r) {
  (“Agencies”, “phone”, r.oid)
}
```

Lineage in Links

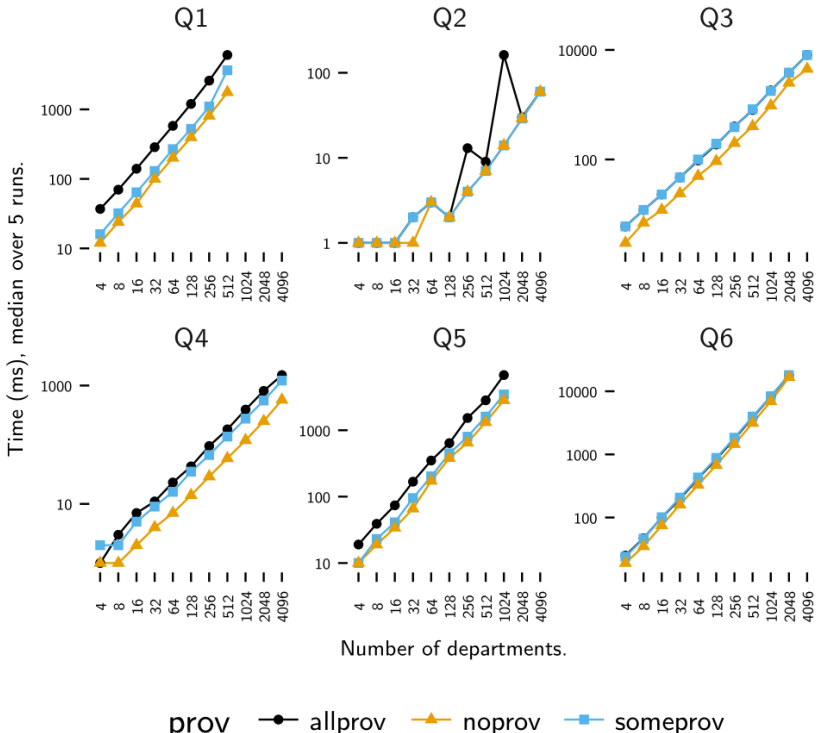
- No special type, **lineage** keyword triggers query rewriting
- Query result changes from [*A*] to [(data: *A*, prov: [(String, Int)])]
- Initial annotations on table with a view
- Add input's annotations to the body's annotations

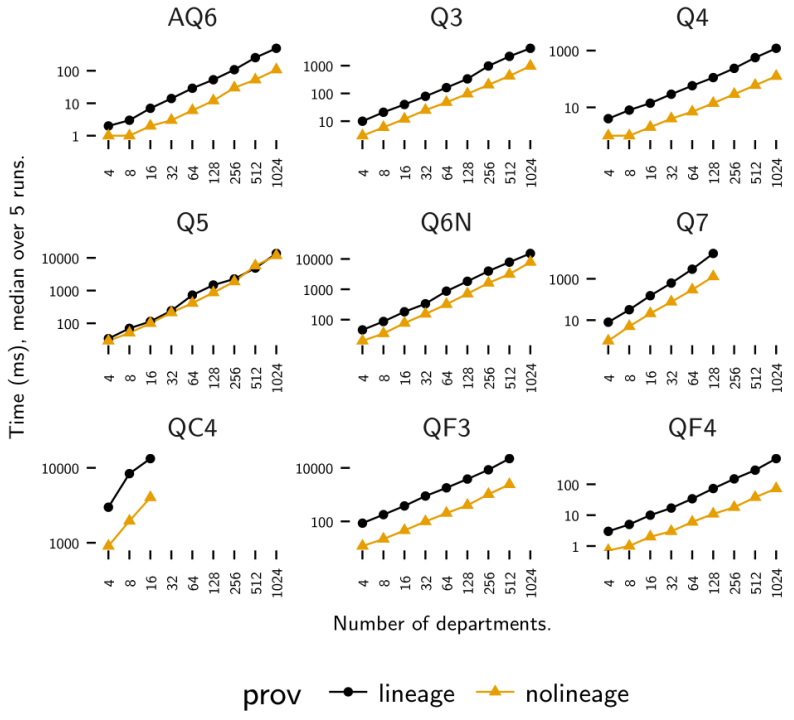
⟦**for** (x <-- N) M⟧ = **for** (y <-- ⟦N'⟧)
for (z <-- ⟦ M'⟧ *[y.data/x]*)
[(data = z.data, prov = y.prov ++ z.prov)]

- Need every function twice, for use within and outwith **lineage** blocks

Performance

- Slowdown for where-provenance: 1.2-2.8x
- Slowdown for lineage: 1.3-7.6x
- Comparable to calculating provenance on the database





Conclusions

Building on language-integrated query technology we can build provenance tracking into the programming language

Type-safe handling of provenance-annotated data

No need for database plugins – works with any plain SQL database

Next steps: richer queries, other forms of provenance, provenance for programming language values, other host languages, …

```
query {
  for (a <- fun() { for (a <-- table “Agencies” …)
                        [(oid = a.oid,
                          name = (data = a.name,
                                  prov = (“Agencies”, “name”, a.oid)),
                          based_in = a.based_in,
                          phone = (data = a.phone,
                                   prov = (“Agencies”, “phone”, a.oid)))]}())
  for (e <- fun() { for (e <-- table “ExternalTours” …) … }())
  where (a.name.data == e.name.data
        && e.type.data == “boat”)
    [(name = e.name,
      phone = a.phone)]
}
```

```
SELECT   e.name              AS name_data,
         'ExternalTours'     AS name_prov_1,
         'name'              AS name_prov_2,
         e.oid               AS name_prov_3,
         a.phone             AS phone_data,
         'Agencies'          AS phone_prov_1,
         'phone'             AS phone_prov_2,
         a.oid               AS phone_prov_3
FROM Agencies AS a,
       ExternalTours AS e
WHERE a.name = e.name
  AND e.type = 'boat'
```